(19)日本国特許庁（ＪＰ）　(12) **公表特許公報（Ａ）**　(11)特許出願公表番号
特表2001－517024
（P2001－517024A）
(43)公表日　平成13年10月2日(2001.10.2)

(51)Int.Cl.[7]	識別記号	ＦＩ		ﾃｰﾏｺｰﾄﾞ(参考)
Ｈ０４Ｌ　12/56		Ｇ０６Ｆ　17/30		４１９Ａ
Ｇ０６Ｆ　17/30	４１９	Ｈ０４Ｌ　11/20		１０２Ｄ

審査請求　未請求　　予備審査請求　有　　（全 46 頁）

(21)出願番号　特願2000－512323(P2000－512323)
(86)（22)出願日　平成10年５月11日(1998.5.11)
(85)翻訳文提出日　平成12年３月15日(2000.3.15)
(86)国際出願番号　ＰＣＴ／ＳＥ９８／００８５４
(87)国際公開番号　ＷＯ９９／１４９０６
(87)国際公開日　平成11年３月25日(1999.3.25)
(31)優先権主張番号　９７０３３３２－８
(32)優先日　平成９年９月15日(1997.9.15)
(33)優先権主張国　スウェーデン（ＳＥ）

(71)出願人　エフィシャント　ネットワーキング　アクティエボラーグ
スウェーデン国，エス・977　75　リュレオ，アウロルム　2
(72)発明者　ブロドレック，アンドレイ
スロベニア共和国，リュブルヤーナ　1000，ゴツカ　6
(72)発明者　デゲルマーク，ミカエル
スウェーデン国，エス－977　52　リュレオ，アシステントベーゲン　322
(74)代理人　弁理士　石田　敬　（外４名）

最終頁に続く

(54)【発明の名称】　高速ルーティング・ルックアップのための方法とシステム

(57)【要約】
IPデータグラムをどこに転送するか決定するための、ネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含むルーティング・テーブル中のIPルーティング・ルックアップの方法において、ルーティング・テーブルの表示が、全てのルーティング・テーブル・エントリのプレフィックスによって規定される完全プレフィックス木（７）の形態で保存される。さらに、現在の深さ（Ｄ）でのプレフィックス木（７）のカットのデータを含むビット・ベクトル（８）の表示と、ネクストホップ・テーブル及びネクスト・レベル・チャンクへの索引を含むポインタの配列とが保存される。前記ビット・ベクトル（８）がビット・マスクに分割され、ビット・マスクの表示がマップテーブルに保存される。その後、各々行索引をマップテーブルとポインタ・オフセットに符号化する符号語の配列と、ベース・アドレスの配列とが保存される。最後に、ルックアップが行われる。

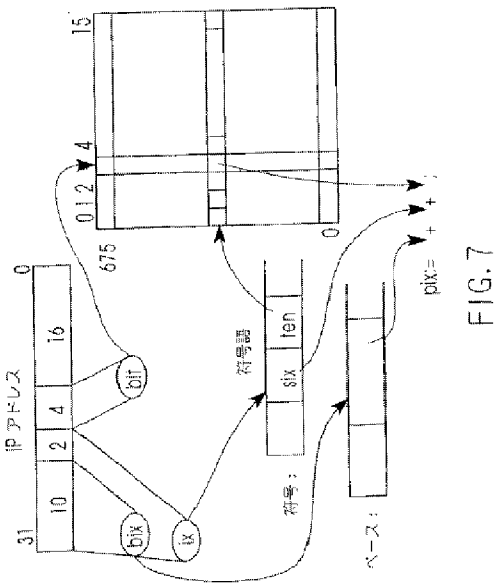

FIG. 7

【特許請求の範囲】

【請求項１】　IPデータグラムをどこに転送するか決定するための、ネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含むルーティング・テーブル中のIPルーティング・ルックアップの方法であって、

各ノードが子を有さないかまたは２つの子を有するかの何れかであり、追加された全ての子が、ネクストホップ情報を有する最も近い先祖と同じネクストホップ情報か、またはこうした先祖が存在しない場合規定されないネクストホップを有する葉であるように完成される、全てのルーティング・テーブル・エントリのプレフィックスによって規定される、完全プレフィックス木（７）の形態での前記ルーティング・テーブルの表示を記憶手段に保存するステップと、

現在の深さ（Ｄ）の可能なノード毎に１ビットを有する前記現在の深さの前記プレフィックス木（７）のカットのデータを含むビット・ベクトル（８）の表示を前記記憶手段に保存するステップであって、その際前記プレフィックス木（７）中にノードが存在する場合前記ビットが設定されるステップと、

ポインタの配列と、純粋ヘッドの場合前記ネクストホップ・テーブルへの索引と、根ヘッドの場合ネクスト・レベル・チャンクへの索引とを前記記憶手段に保存するステップと、

前記ビット・ベクトル（８）をある長さのビット・マスクに分割するステップと、

マップテーブル中の可能な前記ビット・マスクの表示を前記記憶手段に保存するステップと、

各々行索引を前記マップテーブルとポインタ・オフセットに符号化する符号語の配列を前記記憶手段に保存するステップと、

ベース・アドレスの配列を前記記憶手段に保存するステップと、

符号語の前記配列中の前記IPアドレスの第１索引部分(ix)に対応する位置の符号語にアクセスするステップと、

前記IPアドレスの列索引部分(bit) と前記マップテーブル中の前記符号語の行索引部分(ten) とに対応する位置のマップテーブル・エントリ部分にアクセスす

るステップと、

ベース・アドレスの前記配列中の前記IPアドレスの第２索引部分(bix) に対応する位置のベース・アドレスにアクセスするステップと、

前記ベース・アドレス・プラス前記符号語のポインタ・オフセット(six) プラス・ポインタの前記配列中の前記マップテーブル・エントリ部分に対応する位置のポインタにアクセスするステップとを特徴とする方法。

【請求項２】　IPデータグラムをどこに転送するか決定するためのネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含むルーティング・テーブル中のIPルーティング・ルックアップのためのシステムであって、各ノードが子を有さないかまたは２つの子を有するかの何れかであり、追加された全ての子が、ネクストホップ情報を有する最も近い先祖と同じネクストホップ情報か、またはこうした先祖が存在しない場合規定されないネクストホップを有する葉である、全てのルーティング・テーブル・エントリのプレフィックスによって規定される、完全プレフィックス木（７）の形態でのルーティング・テーブルと、

現在の深さ（Ｄ）の可能なノード毎に１ビットを有する前記現在の深さの前記プレフィックス木（７）のカットのデータを含むビット・ベクトル（８）の表示であって、その際前記プレフィックス木（７）中にノードが存在する場合前記ビットが設定される表示と、

ポインタの配列と、純粋ヘッドの場合前記ネクストホップ・テーブルへの索引と、根ヘッドの場合ネクスト・レベル・チャンクへの索引と、

ある長さのビット・マスクに分割された前記ビット・ベクトル（８）と、

可能な前記ビット・マスクの表示を含むマップテーブルと、

各々行索引を前記マップテーブルとポインタ・オフセットに符号化する符号語の配列と、

ベース・アドレスの配列とを特徴とするシステム。

【発明の詳細な説明】

【０００１】

発明の分野

本発明は一般に、IPデータグラムをどこに転送すべきかを決定する、ネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含む、ルーティング・テーブル中のIPルーティング・ルックアップの方法とシステムに関する。

【０００２】

従来技術の説明

インターネットはネットワークの相互接続された集合であり、そこでは構成要素であるネットワークは各々そのアイデンティティを保持し、多数のネットワーク相互の通信のためには専用の機構が必要である。構成要素であるネットワークはサブネットワークと呼ばれる。

【０００３】

インターネット中の各サブネットワークはそのサブネットワークに接続された装置間の通信をサポートする。さらに、サブネットワークは、網間接続ユニット（IWU）と呼ばれる装置によって接続される。

個々のIWU は、似ている、あるいはそうでもない２つのネットワークを接続するために使用されるルータである。このルータは、各ルータとネットワークの各ホストに存在するインターネット・プロトコル（IP）を利用する。

【０００４】

IPは局間のコネクションレスまたはデータグラム・サービスを提供する。

ルーティングは一般に、可能な宛先ネットワークの各々について、IPデータグラムを送信すべき次のルータを示すルーティング・テーブルを各局及びルータに維持することで達成される。

IPルータはルーティング・テーブルでルーティング・ルックアップを行い、どこにIPデータグラムを転送するかを決定する。この操作の結果は宛先方向の経路上のネクストホップである。ルーティング・テーブル中のエントリは概念上、関連ネクストホップ情報を有する任意長プレフィックスである。ルーティング・ル

ックアップは、最長一致プレフィックスを有するルーティング・エントリを見出さなければならない。

【０００５】

IPルーティング・ルックアップは本質的に低速で複雑だったので、従来技術のソリューションによる操作は、それの使用を回避する技術の普及につながった。IPより下の様々なリンク層スイッチング技術、IP層バイパス法（コンピュータ通信会議(IEEE Infocom)会報、カリフォルニア州サンフランシスコ、１９９６年３月、ギガビット・ネットワーク・ワークショップ会報、ボストン、１９９５年４月、及びACM SIGCOMM ’９５、４９～５８ページ、マサチューセッツ州ケンブリッジ、１９９５年８月で開示）及びATM のような仮想回線技術に基づく代替ネットワーク層の開発は、ある程度IPルーティング・ルックアップを回避しようという願望の結果である。

【０００６】

IPレベルより下のスイッチング・リンク層とフローまたはタグ・スイッチング・アーキテクチャの使用によって、複雑性と冗長性がネットワークに追加される。

最新のIPルータ設計はキャッシュ技術（caching technique)を使用しており、そこでは最近使用された宛先アドレスのルーティング・エントリがキャッシュに保持される。この技術は、トラフィックに十分な局所性が存在するので、キャッシュ・ヒット率が十分に高く、ルーティング・ルックアップの費用がいくつかのパケットにわたって分担されるということに依存している。このキャッシュ法は従来良好に動作した。しかし、現在のインターネットの急速な発達によって必要なアドレス・キャッシュのサイズが増大するに連れて、ハードウェア・キャッシュは不経済なものになることがある。

【０００７】

ルーティング・テーブルの伝統的な実現は、ほぼ３０年前に発明されたデータ構造であるパトリシア木(ACMジャーナル、１５（４）：５１４～５３４、１９６８年１０月で開示）の、最長プレフィックス一致のために修正したバージョンを使用している。

例えばNetBSD 1.2の実現におけるような、ルーティング・ルックアップ目的でのパトリシア木の直接的な実現は、葉と内部ノードのために２４バイトを使用する。４０，０００エントリの場合、木構造だけでほぼ２メガバイトであり、完全な平衡木では、ルーティング・エントリを見出すために、１５あるいは１６のノードを横切らなければならない。

【０００８】

場合によっては、最長一致プレフィックス規則のため、適切なルーティング情報を見出すために追加ノードの横断が必要になることがあるが、これは初期探索によって適切な葉が見出されることが保証されていないからである。パトリシア木のサイズを縮小しルックアップ速度を改善できる最適化が存在する。それにもかかわらず、データ構造は大きく、それを探索するためにはあまりにも高価なメモリ参照が必要である。現在のインターネット・ルーティング・テーブルは大きすぎてオンチップ・キャッシュには収まらないし、DRAMのオフチップ・メモリ参照は低速すぎて必要なルーティング速度をサポートすることはできない。

【０００９】

完全なルーティング・ルックアップを回避することでIPルーティング性能を改善する初期の作業（コンピュータ通信会議(IEEE Infocom)会報、ルイジアナ州ニューオーリンズ、１９８８年３月で開示）によって見出されたところでは、小さな宛先アドレス・キャッシュによってルーティング・ルックアップ性能は少なくとも６５パーセント改善できる。９０パーセントを越えるヒット率を得るために必要なスロットは１０未満であった。このような小さな宛先アドレス・キャッシュは、現在のインターネットの大きなトラフィック密度とホスト数には不十分である。

【００１０】

ATM(非同期転送モード）は、接続セットアップ中にアドレスをネットワークに伝える信号プロトコルを有することでルーティング・ルックアップの実行を回避している。仮想回線識別子(VCI）によってアクセスされる転送状態がセットアップ中に接続の経路に沿ったスイッチにインストールされる。ATM セルは、転送状態を有するテーブルへの直接索引またはハッシュ関数へのキーとして使用される

VCI を含んでいる。ルーティングの決定はATM の場合簡単である。しかし、パケット・サイズが４８バイトより大きい場合、さらに多くのATM ルーティングの決定を行う必要がある。

【００１１】

タグ・スイッチング及びフロー・スイッチング（コンピュータ通信会議(IEEE Infocom)会報、カリフォルニア州サンフランシスコ、１９９６年３月で開示）は、ATM 上で動作することを意図する２つのIPバイパス法である。一般的な考え方は、実際のデータ転送を行うリンクレベルATM ハードウェアをIPに制御させるというものである。どのATM 仮想回線識別子を使用し、どのパケットがどのVCI を使用するかをルータ間で一致させる専用プロトコル（コメント要求RFC １９５３、インターネット・エンジニアリング・タスク・フォース、１９９６年５月で開示）が必要である。

【００１２】

同じくIP処理の回避を目標とするもう１つのアプローチはIP／ATM アーキテクチャ（ギガビット・ネットワーク・ワークショップ会報、ボストン、１９９５年４月、及びACM SIGCOMM ’９５会報、４９～５８ページ、マサチューセッツ州ケンブリッジ、１９９５年８月で開示）でなされるが、そこではATM バックプレーンが多数のライン・カードとルーティング・カードを接続する。ルーティング・カードに配置されたIP処理要素がIPヘッダを処理する。パケット・ストリームが到着すると、最初のIPヘッダだけが検討され、その後のパケットは最初のものと同様にルーティングされる。このショートカットの主要な目的は、多数のパケットにわたるIP処理の費用を分担することであると思われる。

【００１３】

IPルータ設計はIBM ルータ（高速ネットワークジャーナル、１（４）：２８１～２８８、１９９３年で開示）の場合のように専用ハードウェアを使用してIP処理を行うこともある。これは柔軟性のないソリューションとなる。IPフォーマットまたはプロトコルに何らかの変化があるとこの設計は無効となる。ソフトウェアの柔軟性と汎用プロセッサの急速な性能の向上によってこのソリューションは好適なものになる。ハードウェアによるもう１つのアプローチは、CAM を使用し

てルーティング・ルックアップを行うことである（コンピュータ通信会議(IEEE Infocom)会報、第３巻、１３８２～１３９１ページ、サンフランシスコ、１９９３年で開示）。これは高速ではあるが高価なソリューションである。

【００１４】

BBN は現在、転送エンジンとして汎用プロセッサを使用する１対のマルチギガビット・ルータを製造中である。今までのところ発表された情報はほとんどない。しかし、この計画は、転送エンジンとしてアルファ・プロセッサ(Alpha processor）を使用し、全てのIP処理をソフトウェアで行うものと思われる。出版物ギガビット・ネットワーキング、マサチューセッツ州レディング、アディソンーウェズリー社、１９９３年が示すところによれば、ルート・キャッシュでのヒットを想定すれば、わずか２００の命令でIP処理を行うことが可能である。アルファの２次キャッシュは宛先アドレスの大規模LRU(最低使用頻度）キャッシュとして使用される。この方式はトラフィック・パターンの局所性を想定している。局所性が低い場合キャッシュ・ヒット率が低くなりすぎ、性能が犠牲になることがある。

【００１５】

発明の概要

従って、ギガビット速度までの各IPパケットについて完全ルーティング・ルックアップを行う改善されたIPルーティング・ルックアップの方法とシステムを提供することが本発明の目的であるが、この方法とシステムは上記で言及された欠点を克服するものである。

【００１６】

さらに別の目的は、従来のマイクロプロセッサによるルーティング・ルックアップ速度を向上させることである。

もう１つの目的は、転送テーブルにおけるルックアップ時間を最小化することである。

本発明のもう１つのさらに別の目的は、従来のマイクロプロセッサのキャッシュに完全に収まるデータ構造を提供することである。

【００１７】

その結果、メモリ・アクセスは、データ構造が、例えば比較的低速なDRAMからなるメモリに存在する必要のある場合より何倍も高速になる。

これらの目的は本発明によるIPルーティング・ルックアップの方法とシステムによって得られるが、このシステムは、非常に小型の形態で大規模なルーティング・テーブルを表すことができ、わずかなメモリ参照を使用して高速で探索できるデータ構造である。

【００１８】

本発明をより詳細に説明し、本発明の利点と特徴を説明するため、好適実施態様が以下詳細に説明され、添付の図面が参照される。

発明の実施の態様

図１を参照すると、ルータ設計は、多数のネットワーク・インバウンド・インタフェース１、ネットワーク・アウトバウンド・インタフェース２、転送エンジン３、及びネットワーク・プロセッサ４を備えているが、これらは全て接続ファブリック５によって相互接続されている。インバウンド・インタフェース１は接続ファブリック５を通じてパケット・ヘッダを転送エンジン３に送信する。一方転送エンジン３はパケットをどの出力インタフェース２に送信すべきかを決定する。この情報はインバウンド・インタフェース１に返送され、そこからパケットがアウトバウンド・インタフェース２に転送される。転送エンジン３の唯一のタスクはパケット・ヘッダを処理することである。ルーティング・プロトコルへの関与、リソースの確保、特別な注意を必要とするパケットの処理、及び他の管理動作といった他の全てのタスクはネットワーク・プロセッサ４によって処理される。

【００１９】

各転送エンジン３は、ネットワーク・プロセッサ４からダウンロードされ、転送エンジン３中の記憶手段に保存されたルーティング・テーブルのローカル・バージョンである転送テーブルを使用してルーティングの決定を行う。ルーティング・アップデートの度に新しい転送テーブルをダウンロードする必要はない。ルーティング・アップデートは頻繁であるが、ルーティング・プロトコルは収束するため若干の時間を要するので、転送テーブルはそれほど陳腐化せず、せいぜい

１秒に１回程度しか変更する必要はない（スタンフォード大学高速ルーティング及びスイッチングに関するワークショップ、１９９６年１２月、http://tiny-tera.stanford.edu/Workshop__Dec96 ／で開示）。

【００２０】

ネットワーク・プロセッサ４は、転送テーブルの高速アップデートと高速生成のために設計された動的ルーティング・テーブルを必要とする。他方、転送テーブルはルックアップ速度について最適化することができ、動的である必要はない。

ルックアップ時間を最小化するために、ルックアップの期間中必要なメモリ・アクセスの数と、データ構造のサイズという２つのパラメータを転送テーブルのデータ構造において最小化しなければならない。

【００２１】

メモリ・アクセスは比較的低速で普通ルックアップ手順のボトルネックとなるため、ルックアップの期間中必要なメモリ・アクセスの数を減少させることは重要である。データ構造は十分に小さくすることができれば、従来のマイクロプロセッサのキャッシュに完全に収まる。これは、パトリシア木の場合のように、データ構造が比較的低速なDRAMからなるメモリに存在する必要がある場合より何桁も高速になることを意味する。

【００２２】

転送テーブルがキャッシュに完全に収まらないばあいでも、テーブルの大部分がキャッシュに存在することができれば有益である。トラフィック・パターンの局所性によってデータ構造の最も頻繁に使用される部分がキャッシュに保持されるので、大部分のルックアップが高速になる。さらに、少量の必要な外部メモリとして高速SRAMを使用することが実行可能になる。SRAMは高価であり、高速であるほどさらに高価になる。費用が一定の場合、SRAMは必要量が少ない方が高速である。

【００２３】

第２の設計目標として、費用のかかる命令と面倒なビット抽出操作を回避するため、データ構造は、ルックアップの期間中に必要な命令が少なく、できる限り

エンティティを自然に整列した状態に保持できるものであるべきである。

データ構造に関する定量的設計パラメータを決定するため、以下説明されるように、多数の大規模ルーティング・テーブルがこれまで検討されている。これらのテーブルに存在する別個のルーティング・エントリは４０，０００とかなり少ない。ネクストホップが同一であれば、残りのルーティング情報も同じであるので、同じネクストホップを指定する全てのルーティング・エントリはルーティング情報を共有できる。ルータのルーティング・テーブル中の別個のネクストホップの数は、１つのホップで到達できる他のルータまたはホストの数によって制限されるので、大規模バックボーン・ルータの場合でもこの数が小さいのは驚くべきことではない。しかし、ルータが例えば大規模ATM ネットワークに接続されている場合、ネクストホップの数はもっと多いことがある。

【００２４】

この実施態様では、転送テーブル・データ構造は2^{14}すなわち１６Ｋの個別ネクストホップに対応するよう設計されているが、これは大部分の場合十分である。別個のネクストホップが２５６より少ない場合、ネクストホップ・テーブルへの索引は１つのバイトに保存できるので、ここで説明される転送テーブルは、別の実施態様で占有する空間がかなり少なくなるように修正できる。

【００２５】

転送テーブルは本質的に３つのレベルを有する木である。１つのレベルの探索には１～４のメモリ・アクセスが必要である。従って、メモリ・アクセスの最大数は１２である。しかし、従来のルーティング・テーブルではルックアップの大多数が必要とするのは１～２レベルだけなので、メモリ・アクセス数の大部分は８以下である。

【００２６】

データ構造を理解する目的で、図２に示される、IPアドレス空間全体に広がる二分木６を想像のこと。その深さは３２であり、葉の数は2^{32}であるが、これは可能な各IPアドレスについて１つである。ルーティング・テーブル・エントリのプレフィックスはあるノードを末端とする木の中の経路を規定する。そのノードに根付いた部分木の全てのIPアドレス（葉）はそのルーティング・エントリによ

ってルーティングされる。この方法で各ルーティング・テーブル・エントリは同一のルーティング情報を有するIPアドレスの範囲を規定する。

【００２７】

いくつかのルーティング・エントリが同じIPアドレスを対象にしている場合、最長一致の規則が適用される。それによれば、あるIPアドレスについて、最長一致プレフィックスを有するルーティング・エントリが使用される。この状況は図３に示されている。ルーティング・エントリｅ１は、範囲ｒ中のアドレスについてｅ２によって隠されている。

【００２８】

転送テーブルは、全てのルーティング・エントリがまたがる二分木６、プレフィックス木７の表示である。プレフィックス木は完全であること、すなわち木の各ノードは２つの子を有するかまたは子を有さない何れかであることが必要である。１つの子を有するノードは２つの子を有するように拡張されなければならない。この形で追加された子は常に葉であり、それらのネクストホップ情報は、ネクストホップ情報を有する最も近い先祖のネクストホップと同じであるか、またはそのような先祖が存在しない場合「規定されない」ネクストホップである。

【００２９】

図４に示されるこの手順は、プレフィックス木７中のノードの数を増加させるが、小さな転送テーブルの構築を可能にする。しかし、転送テーブルを構築するために実際にプレフィックス木を構築する必要はない。プレフィックス木が使用されるのは説明を簡単にするためである。転送テーブルは、全てのルーティング・エントリを１回通過する間に構築できる。

【００３０】

ルーティング・エントリの集合は、IPアドレス空間をIPアドレスの集合に分割する。正しいルーティング情報を見出す問題は、インターバル・セット・メンバーシップ問題（SIAMコンピュータジャーナル、１７（１）：１０９３～１１０２、１９８８年１２月で開示）と同様である。この場合、各インターバルはプレフィックス木中のノードによって規定されるので、転送テーブルを圧縮するために使用できる特性を有する。例えば、IPアドレスの各範囲は２の累乗である長さを

有する。

【００３１】

図５に示されるように、データ構造のレベル１は深さ１６までのプレフィックス木を対象とし、レベル２は深さ１７から２４までを対象とし、レベル３は深さ２５から３２までを対象とする。プレフィックス木の一部がレベル１６の下に延びた場合はいつでも、レベル２チャンク(chunk) が木のその部分を記述する。同様に、レベル３のチャンクはプレフィックス木の２４より深い部分を記述する。データ構造のレベルを探索した結果は、ネクストホップ・テーブルへの索引かまたは次のレベルのチャンクの配列への索引である。

【００３２】

例えば、図５のデータ構造のレベル１には、深さ１６でプレフィックス木７のカット(cut) が存在する。カットは深さ１６で可能なノード１つ毎に１ビットのビット・ベクトルに保存される。すなわち、2^{16}ビット＝６４Ｋビット＝８Ｋバイトが必要である。IPアドレスの最初の部分に対応するビットを見出すため、アドレスの上部の１６ビットがビット・ベクトルへの索引として使用される。

【００３３】

深さ１６のプレフィックス木にノードが存在する場合、ベクトル中の対応するビットが設定される。また、木が１６未満の深さに葉を有する場合、その葉の対象となるインターバルの最下位ビットが設定される。他の全てのビットは０である。すなわち、ビット・ベクトル中のビットは、

プレフィックス木がカットの下まで続くことを表す１、根ヘッド(root head)(図６のビット６、１２及び１３）、または、

深さ１６またはそれ未満の葉を表す１、純粋ヘッド(genuine head)（図６のビット０、４、７、８、１４及び１５）、または、

この値が１６未満の深さの葉の対象となる範囲のメンバ（member）であることを意味する０（図６のビット１、２、３、５、９、１０及び１１）である。メンバは前記メンバより小さい最大ヘッドと同じネクストホップを有する。

【００３４】

純粋ヘッドの場合、ネクストホップ・テーブルへの索引を保存しなければなら

ない。メンバは前記メンバより小さい最大ヘッドと同じ索引を使用する。根ヘッドの場合、対応する部分木を表すレベル２チャンクへの索引を保存しなければならない。ヘッド情報は配列に保存された１６ビット・ポインタで符号化される。ポインタの２ビットはそれがどんな種類のポインタかを符号化し、残りの１４ビットはネクストホップ・テーブルまたはレベル２チャンクを含む配列の何れかへの索引である。

【００３５】

適切なポインタを見出すために、ビット・ベクトルは長さ１６のビット・マスクに分割されるが、それらは2^{12}＝４０９６存在する。さらに、配列でのポインタの位置は、ベース索引、６ビット・オフセット及び４ビット・オフセットという３つのエンティティを加算することで得られる。ベース索引プラス６ビット・オフセットによって特定のビット・マスクに対応するポインタがどこに保存されるかが決定される。４ビット・オフセットはポインタの中のどれを検索すべきかを指定する。図７は、これらのエンティティを見出す方法を示す。以下の段落はその手順の詳細説明である。

【００３６】

ビット・マスクは完全なプレフィックス木から生成されるので、１６ビットの全ての組合せが可能なわけではない。長さ２ｎの０でないビット・マスクは、長さｎの２つのビット・マスクか、または値１のビット・マスクの何らかの組合せである。ａ（ｎ）が長さ2^nのありうる０でないビット・マスクの数であるとする。ａ（ｎ）は次の漸化式によって規定される。

【００３７】

ａ（０）＝１、ａ（ｎ）＝１＋ａ（ｎ－１）^{2}

すなわち、長さ１６のありうるビット・マスクの数はａ（４）＋１＝６７８であるが、１が加算されるのは、ビット・マスクが０のこともあるからである。従って各ビット・マスクに対するエントリを有するテーブルへの索引が必要とするのは１０ビットだけである。

【００３８】

このテーブル、すなわちマップテーブルは、ビット・マスク範囲内のビット数

を４ビット・オフセットにマップするために保持される。このオフセットは必要なポインタを見出すためにいくつポインタをスキップするかを指定するので、ビット索引より小さいセット・ビットの数に等しい。これらのオフセットは、ポインタがたまたまどんな値を有しているかということとは無関係に全ての転送テーブルについて同じである。マップテーブルは一定で、一度に全てについて生成される。

【００３９】

ありうるビットマスクは、長さが２の偶数乗であり、同じ２の累乗の倍数であるビット索引で始まるビットのインターバルが１）全ての０を含むか、または２）最下位ビット・セットを有するかの何れかであるという特性を有する。

マップテーブルによって提供されるオフセットへのビットマスクとビットからのマッピングは、プレフィックス木を完全にしマッピングの使用を可能にするプレフィックスの拡張と共に、小さい転送テーブルサイズを達成する鍵である。

【００４０】

実際のビット・マスクは必要ではなく、ビット・ベクトルの代わりに、マップテーブルへの１０ビットの索引プラス６ビット・オフセットからなる１６ビット符号語(cord word) の配列を保持する。６ビット・オフセットは６４までのポインタを対象とするので、４つの符号語毎に１つのベース索引が必要である。最大６４Ｋのポインタが存在しうるので、ベース索引は最大１６ビット（2^{16}＝６４Ｋ）である必要がある。

【００４１】

図７に示される手順では、データ構造の第１レベルを探索するために疑似符号の次のステップが必要であるが、そこでは符号語の配列が符号と呼ばれ、ベース・アドレスの配列がベースと呼ばれ、ixは符号語の配列中のIPアドレスの第１索引部分であり、bit はマップテーブルへのIPアドレスの列索引部分であり、ten はマップテーブルへの符号語の行索引部分であり、bix はベース・アドレスの配列へのIPアドレスの第２索引部分であり、pix は、レベル２チャンクへの索引と共にネクストホップ・テーブルへの索引を含む、ポインタの配列へのポインタである。

【００４２】

ix　：＝IPアドレスの上位１２ビット

bit　：＝IPアドレス符号語の上位１６ビットの下位４：＝符号［ix］

ten　：＝符号語six からの１０ビット：＝符号語からの６ビット

bix　：＝IPアドレスの上位１０ビット

pix　：＝ベース［bix］＋six ＋マップテーブル［ten］［bit］ポインタ：＝レベル１＿ポインタ［pix］

すなわち、数ビットの抽出、配列参照及び加算だけが必要である。配列を索引する際の暗黙の乗算以外、乗算または除算命令は必要ない。

【００４３】

第１レベルを探索するためにアクセスする必要があるのは合計７バイト、すなわち、２バイトの符号語、２バイトのベース・アドレス、マップテーブル中の１バイト（実際には４ビット）、最後に２バイトのポインタである。第１レベルのサイズは符号語配列について８Ｋバイト、ベース索引の配列について２Ｋバイト、プラス多数のポインタである。マップテーブルが必要とする５．３Ｋバイトは３つのレベル全てで共有される。

【００４４】

ビットマスクが０であるかまたは１つのビット集合(bit set) を有する場合、ポインタはネクストホップ・テーブルへの索引でなければならない。こうしたポインタは直接符号語に符号化できるので、マップテーブルはビット・マスク１及び０のためのエントリを含む必要はない。すなわち、マップテーブル・エントリの数は６７６（索引０～６７５）まで減少する。符号語（上記のten)の１０ビットが６７５より大きい場合、符号語はネクストホップ・テーブルへの直接索引を表す。符号語からの６ビットが索引の最下位６ビットとして使用され、(ten－６７６）が索引の上位ビットである。この符号化によって最大（１０２４－６７６）×2^6 ＝２２２７２の索引が可能になるが、これは設計の対象である１６Ｋより多い。この最適化によって、ルーティング・エントリが深さ１２以上にある時３つのメモリ参照が除去され、ポインタ配列のポインタの数がかなり減少する。これは比較及び条件付き分岐による。

【００４５】

マッピングはCプログラミング言語の以下のデータと関数によって示される。

この符号の重要な特徴は、それが提供する個別のビットマスクからオフセットの配列へのマッピングである。

mtable（及びmt）のエントリを入れ替えるといった変化があっても与えられるマッピングは同じである。従って、４つの１６ビット語の代わりに２つの３２ビット語といった、オフセットの配列を表す他の方法もあるだろう。

```
／* マップテーブル *／
#define MAPVECLEN 4
typedef uint16 MAPVEC [MAPVECLEN] ;
typedef MAPVEC MCOMPACT;
MCOMPACT mt [TMAX];
／* mtはルックアップ時に使用されるマップテーブルである。
```

【００４６】

```
   構築中に使用されるmtableから初期化される。 *／
typedef struct mentry  {
                uint16 mask;  ／* １６ビット・パターン（LSB は
                              設定！） *／
                uint16 len;   ／* ８ビットlen （値１～１６） *／
                MAPVEC map ;  ／* ４のグループ中の１６の４ビットオフセッ
                              ト *／
}   MENTRY, *MP;
void mentry2mcompact(MENTRY *from, MCOMPACT *to)／* 「from」からマップ部分を取り、それを「to」に置く *／
 {
  int i;
  for(i=0; i<;MAPVECLEN; i++) {
      (*to) [i]=from->;map [i];
  }
```

```
}
extern void mtable__compact()
／* mtableからmtを初期化する *／
 {
  register int i;
  for(i=0; i<;TMAX; i++)  {
      mentry2mcompact(&;mtable [i], &; mt [i]);
}
}
```

データ構造のレベル２及び３はチャンクからなる。チャンクは高さ８の部分木を対象とし、最大2^8＝２５６ヘッドを含みうる。レベルｎ－１の根ヘッドはレベルｎのチャンクを指す。

【００４７】

イマジナリ・ビット・ベクトルに含まれるヘッドの数によって、チャンクには３つの種類がある。すなわち、１～８のヘッドが存在する場合、チャンクは疎であり、ヘッドの８ビット索引の配列、それに加えて８つの１６ビット・ポインタ、すなわち合計２４バイトによって表される。

９～６４のヘッドが存在する場合、チャンクは稠密である。それは、ベース索引の数以外レベル１と同様に表される。相違点は、６ビット・オフセットが６４ポインタ全てに及ぶため、１６ビット符号語全てについて必要なベース索引は１つだけだということである。合計３４バイト、それに加えてポインタのための１８～１２８バイトが必要である。

【００４８】

６５～２５６のヘッドが存在する場合、チャンクは超稠密である。それはレベル１と同様に表される。１６の符号語と４のベース索引で合計４０バイトとなる。さらに６５～２５６のポインタが１３０～５１２バイトを必要とする。

稠密及び超稠密チャンクは第１レベルと同様に探索される。

疎なチャンクは、８要素用によく合わせた専用二分探索によって探索される。これは線形探索及び汎用二分探索よりかなり高速である。さらに、この探索は、

条件付き移動命令を有するプロセッサ・アーキテクチャ上で従来の条件付き飛び越しを使用せずに実現できる。

```
／* 疎なチャンクのための探索関数 *／
static inline uint16 findsparse(SPARSECHUNK *chu, uint32 val)
／* chu->;vals は８ビット値のソートされた配列０．．７である。
```

【００４９】

```
   chu->;rinfoはポインタの対応する配列０．．７である。
   val は探索キーである
*／
{
 uint8  *p, *q;
 p=q=&;(chu->;vals  [0 ] );
 p+=( *(p-3)>;val)<;<;2;
 p+=( *(p+1)>;val)<;<;1;
 p+=( *p>;val);

 return (chu->;rinfo [p-q ] );
        }
```

稠密及び超稠密チャンクは、説明されたようにレベル１と同様に最適化される。疎なチャンクでは、２つの連続ヘッドのネクストホップが同一な場合それらを併合（merge)し、小さい方で表すことができる。チャンクが疎か稠密かを決定する場合、この併合が考慮されるので、併合されたヘッドの数が８以下の場合チャンクは疎であると考えられる。木を完全にするために追加された葉の多くは順番に発生し、同一のネクストホップを有する。こうした葉に対応するヘッドは併合されて疎なチャンクになる。

【００５０】

この最適化によって、チャンクの分布はさらに大きい稠密なチャンクからさらに小さい疎なチャンクの方向にシフトされる。大きなテーブルの場合、転送テーブルのサイズは通常５～１５パーセント縮小される。

このデータ構造はルーティング・エントリがかなり増大しても対応できる。現在の設計には２つの制限がある。

【００５１】

１．各種類のチャンクの数はレベル当たり2^{14}　１６３８４に制限される。

表１は、これが現在使用されているものより約１６倍大きいことを示している。それでもこの制限を越える場合、ポインタの符号化を変更し索引にさらに余裕を与えるようにするか、またはポインタ・サイズを増大するようにデータ構造を修正できる。

【００５２】

２．レベル２及び３のポインタの数はベース索引のサイズによって制限される。

現在の実現は１６ビット・ベース索引を使用し、３～５の増大係数に対応できる。この限度を越える場合、ベース・ポインタのサイズを３バイトに増大するのが簡単である。チャンク・サイズは稠密チャンクの場合３パーセント、超稠密チャンクの場合１０パーセント増大する。疎なチャンクには影響はない。

【００５３】

このデータ構造が、ルーティング・エントリの数の大きな増加に対応できることは明らかである。そのサイズはルーティング・エントリの数と共にほぼ直線的に増大する。

転送テーブルの性能を調べるために、多数のIPルーティング・テーブルが集められた。インターネット・ルーティング・テーブルは現在インターネット・パフォーマンス・メジャーメント・アンド・アナリシス（IPMA）プロジェクトのウェブサイト(http://www.ra.net/statistics/）で入手可能であり、以前はルーティング・アービタ・プロジェクト(http://www.ra.net/statistics/）で入手可能となっていたが、これは現在終了している。収集されたルーティング・テーブルは、様々な大規模インターネット相互接続点で使用されるルーティング・テーブルの日々のスナップショットである。これらのテーブルのルーティング・エントリの中には多数のネクストホップを含むものもある。この場合、それらの１つが、転送テーブルで使用されるネクストホップとして無作為に選択された。

【００５４】

図８の表１は様々なルーティング・テーブルから構成された転送テーブルに関するデータを示す。各サイトについて、この表は最大転送テーブルを生成したルーティング・テーブルに関するデータと結果を示す。ルーティング・エントリはルーティング・テーブル中のルーティング・エントリの数であり、ネクストホップはテーブル中に見られる個別のネクストホップの数である。葉はプレフィックス木を完全にするために葉が追加された後のプレフィックス木中の葉の数である。

【００５５】

表１の構築時間は、ルーティング・テーブルのメモリ内二分木表示から転送テーブルを生成するために必要な時間を示す。時間はDEC OSF1を実行する３３３ＭＨｚアルファ２１１６４で測定された。次の列は生成されたテーブルでの疎、稠密及び超稠密チャンクの合計数を示し、それに、データ構造の最下位レベルのチャンクの数が続く。

【００５６】

表１から、新しい転送テーブルが急速に生成できることが明らかである。１Ｈｚの再生周波数では、消費されるアルファの能力は１０分の１未満である。上記で説明されたように、１Ｈｚより高い再生周波数は必要ない。

表１のさらに大きなテーブルはアルファの９６Ｋバイト２次キャッシュには完全に収まらない。しかし、２次キャッシュに収まらない部分のための第３レベル・キャッシュに少量の超高速SRAMを有し、２次キャッシュのミスの費用を低減することは実行可能である。トラフィック・パターンの局所性によって、大部分のメモリ参照は２次キャッシュへのものとなる。

【００５７】

観察された興味深い点は、これらのテーブルのサイズが、配列中の全てのプレフィックスをちょうど保存するために要するものと同等だということである。さらに大きなテーブルでは、プレフィックス毎に必要なのはわずか５．６バイトに過ぎない。これらのバイトの半分以上はポインタによって消費される。スプリント・テーブル(Sprint table)では、３３４６９のポインタが存在し、６５Ｋバイ

ト以上の記憶装置を必要とする。ポインタの数を減らすことで転送テーブルのサイズをさらに縮小できることは明らかである。

【００５８】

ルックアップ・ルーチンに関する測定は、GNU C コンパイラgcc(gnu cc. の使用と移植マニュアル、フリー・ソフトウェア・ファウンデーション、１９９５年１１月、ISBN　１－８８２１１４－６６－３で開示）によってコンパイルされたＣ関数上でなされる。報告された時間には、ネクストホップ・テーブルへの関数呼び出しまたはメモリ・アクセスは含まれない。gcc は、最悪の場合データ構造の１レベルを探索するために約５０のアルファ命令を使用する符号を生成する。ペンティアム・プロでは、gcc は最悪の場合レベル毎に３５から４５の命令を使用する符号を生成する。

【００５９】

以下のＣ符号関数は、測定で使用されるルックアップ関数の符号を示す。

レベル１符号語配列はint1と呼ばれ、ベース索引配列はbase1 と呼ばれる。レベル１のポインタは配列htab1 に保存される。

チャンクは配列cis 、cid 、ciddに保存され、ここで i はレベルである。

稠密及び超稠密チャンクに関するベース・アドレスとポインタはそれぞれ配列baseid、baseidd 、及びhtabid、htabidd に保存される。

```
／* 転送テーブルのためのルックアップ関数 *／
#include "conf.h"
#include "forward.h"
#include "mtentry.h"
#include "mtable.h"
#include "bit2index.h"
#define TABLE __LOOKUP
#include "sparse.h"
#include "timing.h"
#include "lookup.h"
／* これらのマクロは、ipが３２ビット符号なしint(uint32) である時だけ動作
```

```
する */
#define EXTRACT(start, bits, ip) (((ip)<;<;(start))>;>;(32-(bits)))
#define GETTEN(m) (((m)<;<;22)>;>;22)
#define GETSIX(m) ((m)>;>;10)
#define bit2o(ix ,bit)
((mt [(ix)] [(bit)>;>;2] >;>;(((bit)&;0x3)<;<;2))&;0xf)
/* ルックアップ(ipaddr)--ipaddrに関するルーティング・テーブル・エントリ
への索引 */
unsigned int lookup(uint32 ipaddr)
 {
uint32 ix;        /* 符号語配列への索引 */
uint32 code;      /* １６ビット符号語 */
int32 diff;       /* TMAXとの差 */
uint32 ten;       /* mtへの索引 */
uint32 six;       /* 符号の６つの「余分の」ビット */
int32 nhop;       /* ネクストホップへのポインタ */
uint32 off;       /* ポインタ・オフセット */
uint32 hbase;     /* ハッシュ・テーブルのためのベース */
uint32 pntr;      /* ハッシュ・テーブル・エントリ */
int32 kind;       /* pntrの種類 */
uint32 chunk;     /* チャンク索引 */
uint8 *p,*q;      /* 探索疎へのポインタ */
uint32 key;       /* 疎の探索キー */
/*  *HERE* からタイミングを取る   */
```

【００６０】

【表１】

```
ix = EXTRACT(0,12,ipaddr);
code = intl[ix];
ten = GETTEN(code);
six = GETSIX(code);

if ((diff=(ten-TMAX))>=0) {

  nhop = (six | (diff<<6));
  goto end;
}
off = bit2o(ten, EXTRACT(12,4,ipaddr));
hbase = basel[ix>>2];
pntr = htabl[hbase+six+off];

if ((kind = (pntr & 0x3))) {

  chunk = (pntr>>2);

  if (--kind) {
```

【0061】
【表2】

```
if (--kind) {

  key = EXTRACT(16,8,ipaddr);
  p = q = c2s[chunk].vals[0]);

  if ( *(p+3) > key) {
    p += 4;
  };

  while( *p > key) {
    p++;
  };

  pntr = c2s[chunk].rinfo[p - q];

 } else {

  ix = EXTRACT(16,4,ipaddr);
  code = c2dd[chunk][ix];
  ten = GETTEN(code);
  six = GETSIX(code);

  if ((diff=(ten-TMAX))>=0) {

    nhop = (six | (diff<<6));
    goto end;
  }
  hbase = htab2ddbase[(chunk<<2)|(ix>>2)];
  off = bit2o(ten, EXTRACT(20,4,ipaddr));
  pntr = htab2dd[hbase+six+off];
 }
} else {

  ix = EXTRACT(16,4,ipaddr);
  code = c2d[chunk][ix];
```

【0062】

【表3】

```
  ten = GETTEN(code);
  six = GETSIX(code);

  if ((diff=(ten-TMAX))>=0) {

    nhop = (six | (diff<<6));
    goto end;
  }
  hbase = htab2dbase[chunk];
  off = bit2o(ten, EXTRACT(16,4,ipaddr));
  pntr = htab2d[hbase+six+off];
}

if ((kind = (pntr & 0x3))) {

   chunk = (pntr>>2);

   if (--kind) {
     If (--kind) {

       key = EXTRACT(24,8,ipaddr);
       p = q = c3s[chunk].vals[0]);

       if ( *(p+3) > key) {
         p += 4;
       };

       while( *p > key) {
         p++;
       };

       pntr = c3s[chunk].rinfo[p - q];

     } else {
```

【0063】

【表4】

```
                ix = EXTRACT(24,4,ipaddr);
                code = c3dd[chunk][ix];
                ten = GETTEN(code);
                six = GETSIX(code);

                if ((diff=(ten-TMAX))>=0) {

                  nhop = (six | (diff<<6));
                  goto end;
                }
                off = bit2o(ten, EXTRACT(28,4,ipaddr));
                hbase = htab3ddbase[(chunk<<2)|(ix>>2)];
                pntr = htab3dd[hbase+six+off];
              }
            } else {

              ix = EXTRACT(24,4,ipaddr);
              code = c3d[chunk][ix];
              ten = GETTEN(code);
              six = GETSIX(code);

              if ((diff=(ten-TMAX))>=0) {

                nhop = (six | (diff<<6));
                goto end;
              }
              off = bit2o(ten, EXTRACT(28,4,ipaddr));
              hbase = htab3dbase[chunk];
              pntr = htab3d[hbase+six+off];
            }
          }
        }
        nhop = (pntr >> 2);

      end:
```

【００６４】

```
/*  HERE！までタイミングを取る  */
return nhop;
}
```

アルファ及びペンティアム・プロ(Pentium Pro) のクロックサイクル・カウンタの現在の値を読むことが可能である。高い精度でルックアップ時間を測定するためにこの機構が使用された。１クロックチック(clock tick)は２００ＭＨｚで

は５ナノ秒であり、３３３ＭＨｚでは３ナノ秒である。

【００６５】

理想的には、転送テーブル全体がキャッシュに配置され、ルックアップは非擾乱キャッシュ(undisturbed cache) によって行われる。これは専用転送エンジンのキャッシュの挙動をエミュレートする。しかし、測定は従来の汎用ワークステーションで行われたのであって、こうしたシステムでキャッシュ内容を制御することは困難である。キャッシュは、Ｉ／Ｏが行われる時、割込みが発生する時、または他の処理の実行が開始される時常に擾乱される。キャッシュを擾乱せずには、測定データをプリントアウトすることやファイルから新しいIPアドレスを読み出すことも不可能である。

【００６６】

使用される方法は各ルックアップを２回行い、２回目のルックアップのルックアップ時間を測定する。この方法では、最初のルックアップは擾乱キャッシュ(disturbed cache) によってなされ、２回目は全ての必要なデータが最初のルックアップによって１次キャッシュに置かれた状態のキャッシュでなされる。各１組のルックアップの後で測定データがプリントアウトされ新しいアドレスがフェッチされるが、この手順は再びキャッシュを擾乱する。

【００６７】

２回目のルックアップは、データと命令がプロセッサに最も近い１次キャッシュに移動しているため、転送エンジンにおけるルックアップより良好に行われる。ルックアップ時間の上限値を得るために、２次キャッシュへのメモリ・アクセスのために必要な追加時間を測定時間に加算しなければならない。転送テーブルを通る全ての経路を試験するために、完全木への拡張によって追加されたエントリを含む、ルーティング・テーブルの各エントリについてルックアップ時間が測定された。

【００６８】

現実のトラフィック・ミックス(traffic mix) が各ルーティング・エントリへのアクセスに関して均一な確率を有することはありそうにないので、平均ルックアップ時間はこれらの実験からは推論できない。さらに、トラフィック・パター

ンの局所性によってデータ構造の頻繁にアクセスされる部分は１次キャッシュに保持されるので、平均ルックアップ時間は減少する。以下計算される性能値は、全てのメモリ・アクセスは１次キャッシュでミスし、常に最悪の場合の実行時間が発生することを想定しているため控え目なものである。現実のルックアップ速度はもっと高速である。

【００６９】

表１は、データ構造のレベル３ではチャンクが非常に少ないことを示す。このため、ネクストホップを見出すためルックアップの大部分が探索する必要があるのはせいぜい２レベルまでということになる。従って、２次キャッシュへのメモリ・アクセスのための追加時間は最悪の場合の１２ではなく８のメモリ・アクセスについて計算される。もしルックアップの大きな部分がそれらの少数のチャンクにアクセスする場合があれば、それらは１次キャッシュに移動するので、１２全てのメモリ・アクセスはより費用のかからないものになるだろう。

【００７０】

実験はアルファ２１１６４上で、３３３ＭＨｚのクロック周波数で行われた。１サイクルは３ナノ秒である。８Ｋバイト１次データ・キャッシュへのアクセスは２サイクルで完了し、２次９６Ｋバイト・キャッシュへのアクセスには８サイクルが必要である。図９の表２を参照のこと。

図１０は、１月１日からのスプリント・ルーティング・テーブルに関するアルファの場合の２回目のルックアップの期間に経過したクロック・チックの分布を示す。観察された最高速のルックアップは１７クロック・サイクルを必要とする。これは、第１レベルの符号語がネクストホップ・テーブルへの索引を直接符号化する場合である。こうしたルーティング・エントリは非常に少数である。しかし、こうしたルーティング・エントリは各々多くのIPアドレスを対象とするので、実際のトラフィックはこうした宛先アドレスを多く含むことがある。ルックアップには２２サイクルかかるものがあるが、これは前と同じ場合であろう。クロック・サイクル・カウンタが２つの連続命令について読み取られる場合、その差は予想される０ではなく５サイクルのことがあることが、実験によって確認されている。

【００７１】

図１０の次のスパイク(spike) は４１クロック・サイクルにあるが、これは第１レベルにあるポインタがネクストホップ・テーブルへの索引である場合である。伝統的なクラスＢアドレスがこのカテゴリに入る。５２～５３、５７、６２、６７、及び７２チックのスパイクは、疎なレベル２チャンクで１、２、３、４または５の値を調べた後にポインタが見出されることに対応する。７５及び８３チックのスパイクが非常に大きいのは、多くのチックがそれぞれ稠密及び超稠密チャンクを探索する必要があるためである。８３以上でいくつかのチックが観察されるのは、おそらく実行時間の変化のために疎なレベル３チャンクを探索した後で見出されるポインタに対応する。２次キャッシュでのキャッシュの競合、またはルックアップ前のパイプライン及びキャッシュ・システムの状態の差によってこうした変化が起きることがある。１００クロック・サイクルを越える観察値の末尾は、こうした変化かまたはキャッシュ・ミスの何れかによるものである。全てのデータが１次キャッシュにある場合３００ナノ秒あれば十分である。

【００７２】

１次キャッシュと２次キャッシュのデータ・アクセスの差は８－２＝６サイクルである。データ構造の２レベルを探索するのに必要なクロック・サイクルは最悪の場合、図１０に示された場合より８×６＝４８多い。これは、２レベルが十分である時、最悪の場合のルックアップでは最大１００＋４８＝１４８サイクルすなわち４４４ナノ秒が必要となることを意味する。すなわち、アルファは、２次キャッシュ中の転送テーブルで、１秒当たり２２０万のルーティング・ルックアップを行うことができる。

【００７３】

もう１つの実験はペンティアム・プロ上で、２００ＭＨｚのクロック周波数で行われた。１サイクルは５ナノ秒である。１次８Ｋバイト・キャッシュは２サイクルの待ち時間を有し、２５６Ｋバイトの２次キャッシュは６サイクルの待ち時間を有する。表２を参照のこと。

図１１は、アルファ２１１６４の場合と同じ転送テーブルに関するペンティアム・プロの場合の２回目のルックアップの期間に経過したクロック・チックの分

布を示す。クロック・サイクル・カウンタを取り出す（fetch)一連の命令は３３クロック・サイクルを要する。互いの直後に２つの取り出しが発生する場合カウンタ値は３３異なる。この理由で、報告された時間は全て３３減らされている。

【００７４】

観察された最高速のルックアップは１１クロック・サイクルであり、アルファ２１１６４の場合とほぼ同じ速度である。ネクストホップ索引が第１レベルの直後にある場合に対応するスパイクは２５クロック・サイクルで発生する。疎なレベル２チャンクに対応するスパイクは３６～４０クロック・サイクルの範囲に互いに接近して集まっている。ペンティアムの異なったキャッシュ構造は、アルファ２１１６４のキャッシュ構造より線形走査を良好に扱うように思われる。

【００７５】

第２レベル・チャンクが稠密及び超稠密である時、ルックアップはそれぞれ４８及び５０サイクルを必要とする。６９まではいくつかの付加的な不規則スパイクがあるが、それ以上で観察されるものは非常に少ない。全てのデータが１次キャッシュにある場合、ルックアップを行うには６９サイクル（３４５ナノ秒）で十分なことは明らかである。

【００７６】

１次キャッシュと２次キャッシュのアクセス時間の差は２０ナノ秒（４サイクル）である。２つのレベルを調べる必要がある時、ペンティアム・プロの場合のルックアップ時間は最悪で６９＋８×４＝１０１サイクルすなわち５０５ナノ秒である。ペンティアム・プロは２次キャッシュの転送テーブルで１秒当たり少なくとも２００万のルーティング・ルックアップを行うことができる。

【００７７】

本発明が、上記で示した目標と利点を完全に満足する改善されたIPルーティング・ルックアップの方法とシステムを提供することは明らかであろう。本発明は特定の実施態様と共に説明されたが、代替案、修正及び変形は当業技術分野に熟練した者には明らかである。

上記で説明された実施態様ではルックアップ関数はCプログラミング言語で実現されている。プログラミングの技術分野に熟練した者には、他のプログラミン

グ言語も使用できることが明らかであろう。また、ルックアップ関数は標準デジタル設計技術を使用してハードウェアで実現することもできるが、これもハードウェア設計の技術分野に熟練した者には明らかであろう。

【００７８】

例えば、本発明は、アルファ２１１６４またはペンティアム・プロを使用するシステム以外のコンピュータ・システム構成、プレフィックス木をカットする他の方法、木の様々なレベル数、マップテーブルを表す他の方法、及び符号語を符号化する他の方法に適用可能である。

さらに、本発明はファイアウォール・ルーティング・ルックアップにも利用できる。

【図面の簡単な説明】

【図１】

図１は、ルータ設計の概略図である。

【図２】

図２は、IPアドレス空間全体にわたる二分木を示す図である。

【図３】

図３は、IPアドレスの範囲を規定するルーティング・エントリを示す図である。

【図４】

図４は、プレフィックス木を拡張して完全なものにするステップを示す図である。

【図５】

図５は、本発明によるデータ構造の３つのレベルを示す図である。

【図６】

図６は、深さ１６でのプレフィックス木のカットの一部を示す図である。

【図７】

図７は、データ構造の第１レベル探索を示す図である。

【図８】

図８は、様々なルーティング・テーブルから構成された転送テーブル上のデー

タを示す表である。

【図９】

図９は、プロセッサとキャッシュのデータを示す表である。

【図１０】

図１０は、アルファ２１１６４に関するルックアップ時間分布を示すグラフである。

【図１１】

図１１は、ペンティアム・プロに関するルックアップ時間分布を示すグラフである。

【図１】

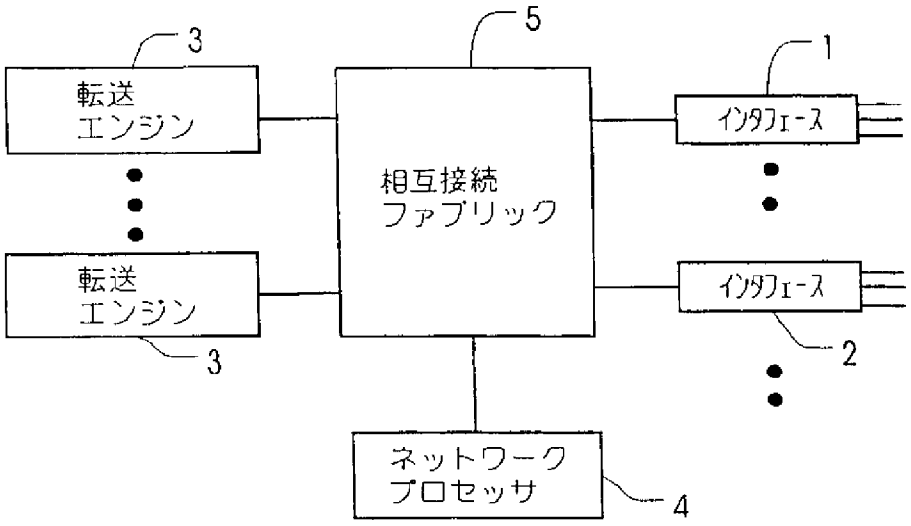

FIG. 1

【図２】

FIG. 2

【図３】

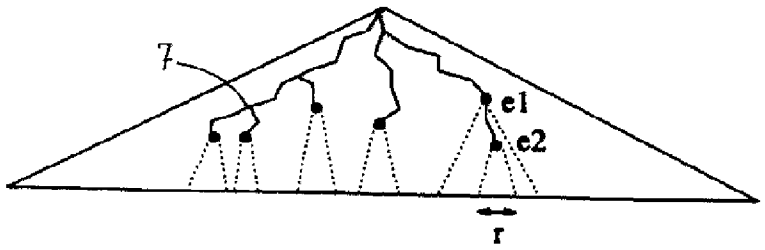

FIG. 3

【図４】

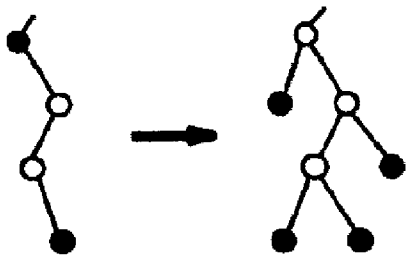

FIG. 4

【図５】

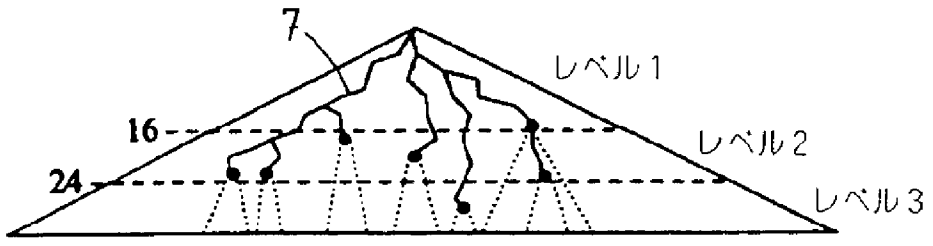

FIG. 5

【図６】

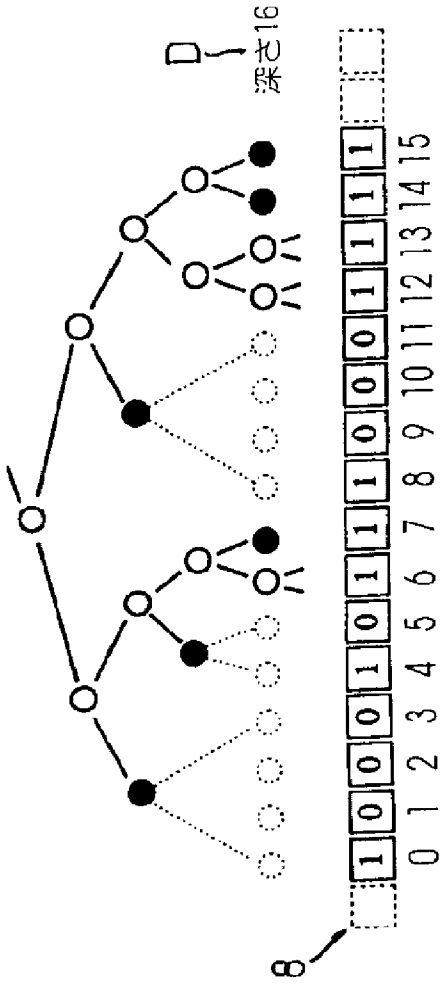

FIG. 6

【図７】

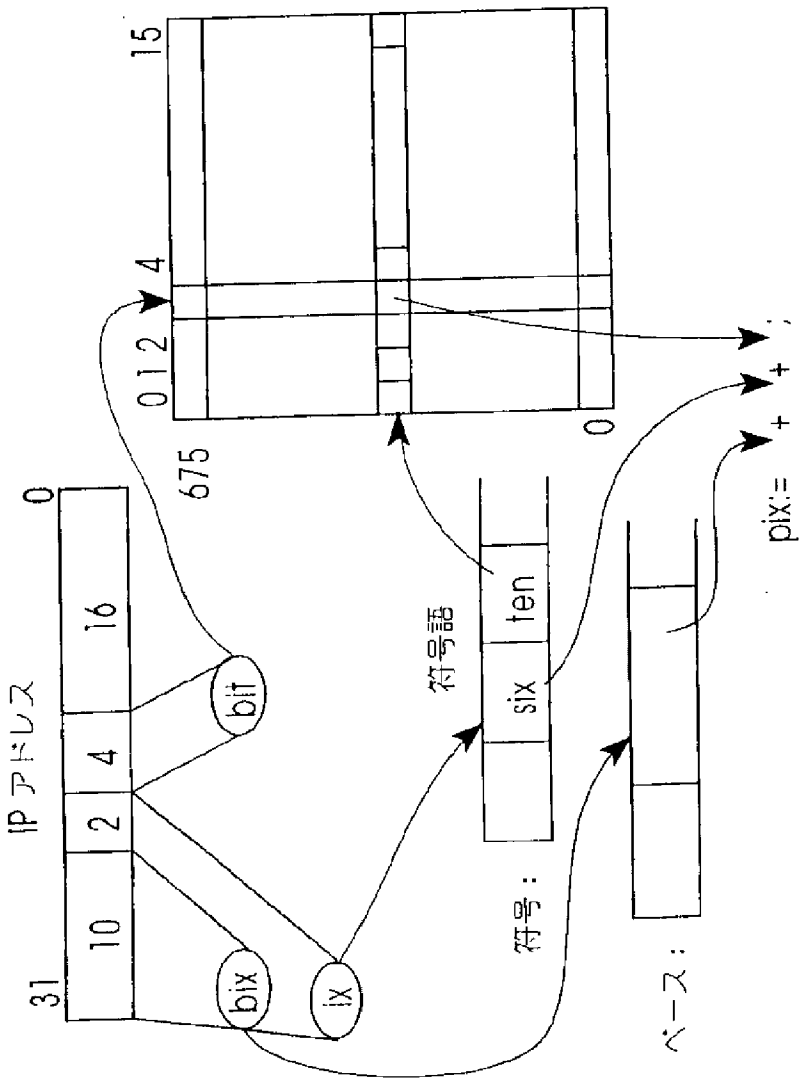
15
4
0 1 2
675
0
IP アドレス
0
16
4
2
10
31
bit
bix
ix
符号語
ten
six
符号：
ベース：
pix:=
+
+

FIG. 7

【図８】

表１

サイト	月日	年	ルーティング エントリー	葉	ネクスト ホップ	サイズ (Kb)	構築 時間	疎な チャンク	稠密 チャンク	超稠密 チャンク	レベル３ チャンク
Mae E	１月９日	'97	32732	58714	56	160	99ミリ秒	1199	587	186	2
Mae E	10月21日	'96	38141	36607	50	148	91ミリ秒	1060	593	149	4
Sprint	１月１日	'97	21797	43513	17	123	72ミリ秒	988	483	98	3
Pac Bell	１月28日	'97	18308	33250	2	99	49ミリ秒	873	357	67	0
MaeW	１月１日	'97	12049	28273	51	86	46ミリ秒	775	312	42	3
AADS	１月４日	'97	1109	5670	12	28	11ミリ秒	320	38	0	2

FIG. 8

【図９】

表２

プロセッサ	クロック サイクル	１次キャッシュ サイズ	１次キャッシュ 待ち時間	２次キャッシュ サイズ	２次キャッシュ 待ち時間	３次キャッシュ サイズ	３次キャッシュ 待ち時間
アルファ21164	3ナノ秒	8 Kバイト	6ナノ秒	96 Kバイト	24ナノ秒	2 Mバイト	2ナノ秒
ペンティアムプロ	5ナノ秒	8 Kバイト	10ナノ秒	256 Kバイト	30ナノ秒		

FIG. 9

【図１０】

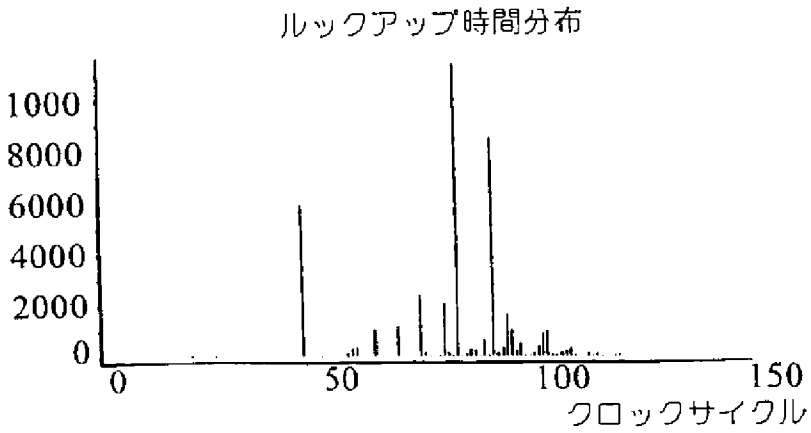

FIG. 10

【図１１】

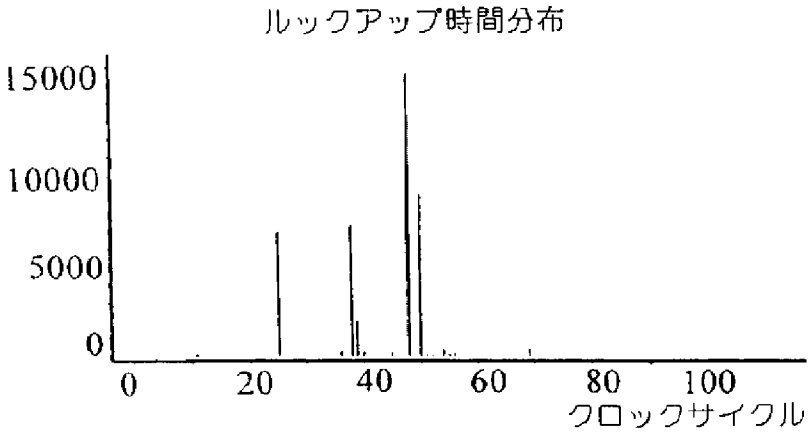

FIG. 11

【手続補正書】特許協力条約第３４条補正の翻訳文提出書
【提出日】平成１２年３月１５日（２０００．３．１５）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】　IPデータグラムをどこに転送するか決定するための、ネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含むルーティング・テーブル中のIPルーティング・ルックアップの方法であって、
各ノードが子を有さないかまたは２つの子を有するかの何れかであり、追加された全ての子が、ネクストホップ情報を有する最も近い先祖と同じネクストホップ情報か、またはそうした先祖が存在しない場合規定されないネクストホップを有する葉であるように完成される、全てのルーティング・テーブル・エントリのプレフィックスによって規定される、完全プレフィックス木（７）の形態での前記ルーティング・テーブルの表示を記憶手段に保存するステップと、
現在の深さ（Ｄ）の可能なノード毎に１ビットを有する前記現在の深さの前記プレフィックス木（７）のカットのデータを含むビット・ベクトル（８）の表示を前記記憶手段に保存するステップであって、その際前記プレフィックス木（７）中にノードが存在する場合前記ビットが設定されるステップと、
純粋ヘッドの場合前記ネクストホップ・テーブルへの索引を含み、根ヘッドの場合ネクスト・レベル・チャンクへの索引を含む、ポインタの第１レベル配列を前記記憶手段に保存するステップと、
前記ビット・ベクトル（８）をある長さのビット・マスクに分割するステップと、
マップテーブル中の可能な前記ビット・マスクの表示を前記記憶手段に保存するステップと、

各々行索引を前記マップテーブルとポインタ・オフセットに符号化する符号語の配列を前記記憶手段に保存するステップと、

ベース・アドレスの配列を前記記憶手段に保存するステップと、

符号語の前記配列中の前記IPアドレスの第１索引部分(ix)に対応する位置の符号語にアクセスするステップと、

前記IPアドレスの列索引部分(bit) と、前記マップテーブル中の前記符号語の行索引部分(ten) とに対応する位置のマップテーブル・エントリ部分にアクセスするステップと、

ベース・アドレスの前記配列中の前記IPアドレスの第２索引部分(bix) に対応する位置のベース・アドレスにアクセスするステップと、

前記ベース・アドレス・プラス前記符号語のポインタ・オフセット(six) プラス・ポインタの前記配列中の前記マップテーブル・エントリ部分に対応する位置のポインタにアクセスするステップとを含む方法において、

マップテーブル中の可能な前記ビット・マスクの表示を前記記憶手段に保存する前記ステップが、

長さが２の偶数乗であり、同じ２の累乗の倍数であるビット索引で始まるビットのどのインターバルも全ての０を含むか、または最下位ビット・セットを有するかの何れかであるという特性を有する可能な前記ビット・マスクの表示を保存するステップを含むことを特徴とする方法。

【請求項２】　前記ポインタがネクストレベル・チャンクへの索引である場合、さらに、純粋ヘッドの場合前記ネクストホップ・テーブルへの索引を含み、根ヘッドの場合第３レベル・チャンクへの索引を含む、ポインタの第２レベル・チャンク配列中のポインタを見出すために、第２レベル・チャンクにアクセスするステップを含むことを特徴とする、請求項１に記載の方法。

【請求項３】　ポインタの前記第２レベル・チャンク配列中の前記ポインタが前記第３レベル・チャンクへの索引である場合、さらに、前記ネクストホップ・テーブルへの索引を含むポインタの第３レベル・チャンク配列中のポインタを見出すために、前記第３レベル・チャンクにアクセスするステップを含むことを特徴とする、請求項２に記載の方法。

【請求項４】　前記チャンクが１～８ヘッドを含む場合、前記チャンクが疎なチャンクであり、前記ヘッドの８ビット索引の配列と８つの１６ビット・ポインタによって表されることと、

前記チャンクが９～６４ヘッドを含む場合、前記チャンクが稠密なチャンクであり、前記第１レベルと同様に表されることと、

前記チャンクが６５～２５６ヘッドを含む場合、前記チャンクが超稠密なチャンクであり、前記第１レベルと同様に表されることとを特徴とする、請求項１～３の何れか１項に記載の方法。

【請求項５】　IPデータグラムをどこに転送するか決定するためのネクストホップ・テーブル中の関連ネクストホップ情報を有する任意長プレフィックスのエントリを含むルーティング・テーブル中のIPルーティング・ルックアップのためのシステムであって、各ノードが子を有さないかまたは２つの子を有するかの何れかであり、追加された全ての子が、ネクストホップ情報を有する最も近い先祖と同じネクストホップ情報か、またはこうした先祖が存在しない場合規定されないネクストホップを有する葉である、全てのルーティング・テーブル・エントリのプレフィックスによって規定される、完全プレフィックス木（７）の形態でのルーティング・テーブルと、

現在の深さ（Ｄ）の可能なノード毎に１ビットを有する前記現在の深さの前記プレフィックス木（７）のカットのデータを含むビット・ベクトル（８）の表示であって、その際前記プレフィックス木（７）中にノードが存在する場合前記ビットが設定される表示と、

純粋ヘッドの場合前記ネクストホップ・テーブルへの索引を含み、根ヘッドの場合ネクストレベル・チャンクへの索引を含む、ポインタの配列と、

ある長さのビット・マスクに分割された前記ビット・ベクトル（８）と、

可能な前記ビット・マスクの表示を含むマップテーブルと、

各々行索引を前記マップテーブルとポインタ・オフセットに符号化する符号語の配列と、

ベース・アドレスの配列とを備える方法において、

可能な前記ビット・マスクが、長さが２の偶数乗であり、同じ２の累乗の倍数

であるビット索引で始まるビットのどのインターバルも全ての０を含むか、または最下位ビット・セットを有するかの何れかであるという特性を有することを特徴とするシステム。

【請求項６】　チャンクの前記ネクストレベルが、純粋ヘッドの場合前記ネクストホップ・テーブルへの索引を含み、根ヘッドの場合第３レベル・チャンクへの索引を含むポインタの第２レベル・チャンク配列によって表される第２レベル・チャンクであることを特徴とする、請求項５に記載のシステム。

【請求項７】　前記第３レベル・チャンクが、前記ネクストホップ・テーブルへの索引を含む、ポインタの第３レベル・チャンク配列によって表されることを特徴とする、請求項６に記載のシステム。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 98/00854

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: H04L 12/56
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EDOC, WPIL, JAPIO, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Mikael Degermark, "TREE ASPECTS OF PACKET FORWARDING IN THE INTERNET", April 1997, (Luleå Tekniska Högskola, Sweden), page 27 - page 44, Doctoralthesis ---	1,2
A	Infocom, Volume 3, 1993, (Morristown, USA) Anthony J. McAuley & Paul Francis, "Fast Routing Table Lookup Using CAMs" ---	1,2
A	IBM Technical Disclosure Bulletin, Volume 36, No 2, February 1993, ., "Memory Organization Scheme for the Implementation of Routing Tables in High Performance IP Routers" page 151 - page 153 --	1,2

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" enter document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 October 1998	25 -10- 1998
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Anders Ströbeck Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 98/00854

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM Disclosure Bulletin, Volume 40, No 3, March 1997, ., "Technique for Performing Generalized Prefix Matches" page 189 - page 200 --	1,2
A	SIAM Journal Computers, Volume 17, No 6, December 1988, Kurt Mehlhorn et al, "A LOWER BOUND ON THE COMPLEXITY OF THE UNION-SPLIT-FIND PROBLEM" page 1093 - page 1102 -- --------	1,2

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

フロントページの続き

(81)指定国　　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＣＹ，ＤＥ，ＤＫ，ＥＳ，ＦＩ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，ＬＵ，ＭＣ，ＮＬ，ＰＴ，ＳＥ)，ＯＡ(ＢＦ，ＢＪ，ＣＦ，ＣＧ，ＣＩ，ＣＭ，ＧＡ，ＧＮ，ＭＬ，ＭＲ，ＮＥ，ＳＮ，ＴＤ，ＴＧ)，ＡＰ(ＧＨ，ＧＭ，ＫＥ，ＬＳ，ＭＷ，ＳＤ，ＳＺ，ＵＧ，ＺＷ)，ＥＡ(ＡＭ，ＡＺ，ＢＹ，ＫＧ，ＫＺ，ＭＤ，ＲＵ，ＴＪ，ＴＭ)，ＡＬ，ＡＭ，ＡＴ，ＡＵ，ＡＺ，ＢＡ，ＢＢ，ＢＧ，ＢＲ，ＢＹ，ＣＡ，ＣＨ，ＣＮ，ＣＵ，ＣＺ，ＤＥ，ＤＫ，ＥＥ，ＥＳ，ＦＩ，ＧＢ，ＧＥ，ＧＨ，ＧＭ，ＧＷ，ＨＵ，ＩＤ，ＩＬ，ＩＳ，ＪＰ，ＫＥ，ＫＧ，ＫＰ，ＫＲ，ＫＺ，ＬＣ，ＬＫ，ＬＲ，ＬＳ，ＬＴ，ＬＵ，ＬＶ，ＭＤ，ＭＧ，ＭＫ，ＭＮ，ＭＷ，ＭＸ，ＮＯ，ＮＺ，ＰＬ，ＰＴ，ＲＯ，ＲＵ，ＳＤ，ＳＥ，ＳＧ，ＳＩ，ＳＫ，ＳＬ，ＴＪ，ＴＭ，ＴＲ，ＴＴ，ＵＡ，ＵＧ，ＵＺ，ＶＮ，ＹＵ，ＺＷ

(72)発明者　カールソン，スバンテ
スウェーデン国，エスー977　53　リュレオ，ニュスティーゲン　4

(72)発明者　ピンク，ステファン
スウェーデン国，エスー165　57　ハースレビー，ビベカ　トロレス　グランド　8